Universidade Federal de Alagoas (UFAL)
Campus Arapiraca

Programação Orientada a Objetos (POO)

05 - Modificadores e encapsulamento

Alexandre de Andrade Barbosa

alexandre.barbosa@arapiraca.ufal.br

Objetivos

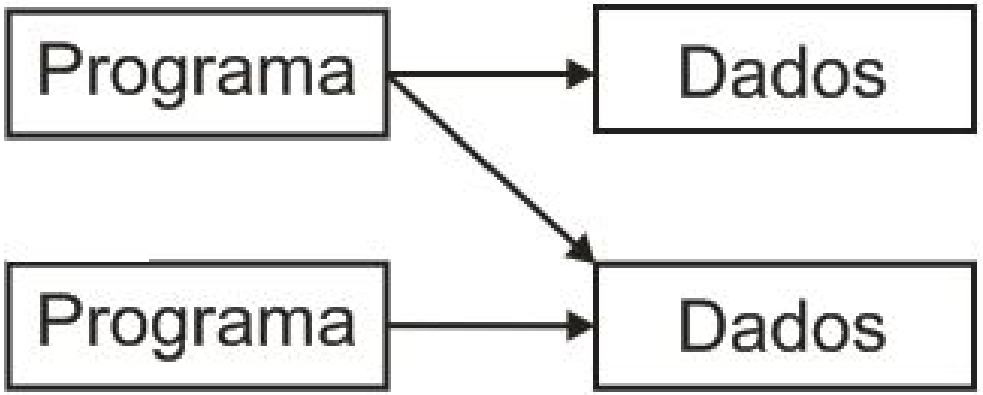
Objetivos

- Conhecer os modificadores existentes na linguagem Java
- Compreender o funcionamento da cada um dos modificadores
- Descrever a implementação do conceito de encapsulamento utilizando modificadores de acesso

Modificadores e encapsulamento

Modificadores

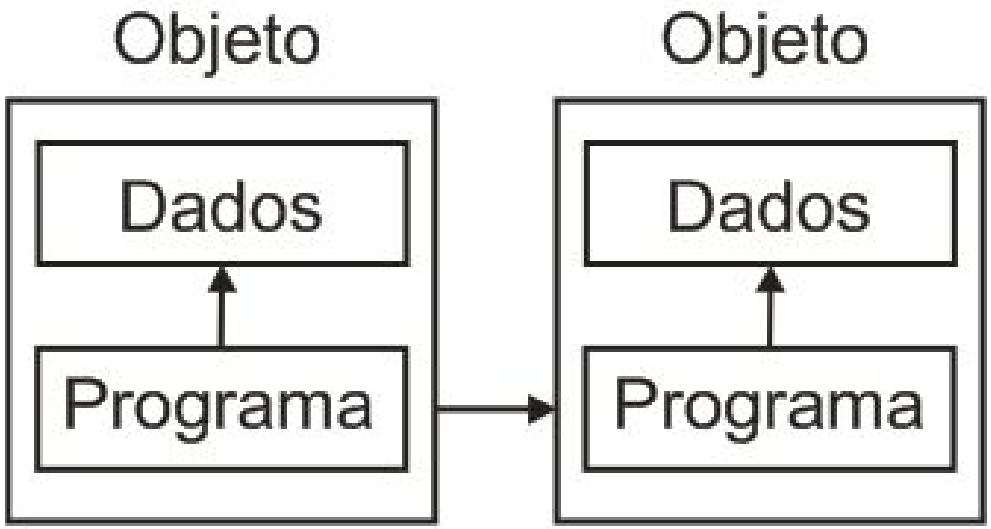
- Um dos principais problemas da programação estruturada é o acesso "descontrolado" a um conjunto de dados.
- Qualquer "programa" pode acessar um conjunto de dados.



Modificadores e encapsulamento

Modificadores

- Modificadores de acesso permitem controlar o acesso a um conjunto de dados
- O desenvolvedor tem controle sobre quais programas podem alterar quais informações.



Modificadores e encapsulamento

Modificadores

- Modificadores alteram a forma de uso de classes, métodos e atributos
- Pode-se descrever modificadores relacionados apenas com a restrição de acesso aos métodos e atributos de uma classe
- São modificadores em Java: private, protected, package, public, static, final, abstract, synchronized, transient, strictfp, volative e native
- Os sete primeiros modificadores são mais comuns, apenas estes serão descritos detalhadamente

Modificadores e encapsulamento

Modificadores

Modificador: public

- Aplicável: classe, método e atributo
- Descrição: torna o elemento (classe, método ou atributo) acessível para qualquer classe

Modificadores e encapsulamento

Modificadores

```
1  package aula05.acesso.garagem; // local da classe
2
3  public class Carro {
4      public String marca; // atributo publico
5      public String modelo; // atributo publico
6      public float velocidade; // atributo publico
7      // construtores
8      public Carro() {}
9      public Carro(String marca, String modelo, float
           velocidade) {
10         this.marca = marca;
11         this.modelo = modelo;
12         this.velocidade = velocidade;
13     }
14
15     public String toString() { // metodo publico
16         return "Carro [marca=" + marca + ", modelo=" + modelo
               + ", velocidade="+ velocidade + "]";
17     }
18
19 }
```

Modificadores e encapsulamento

Modificadores

```
1  package aula05.acesso.oficina; // local da classe, diferente
       da classe Carro
2
3  import aula05.acesso.garagem.Carro; // necessita import para
       ver a classe
4
5  public class Mecanico {
6      // classe acessível pois classe é 'public'
7      Carro carro;
8
9      public void exemplo() {
10         // construtor acessível pois é 'public'
11         carro = new Carro("fiat", "mille");
12         // atributo acessível pois é 'public'
13         System.out.println(carro.modelo);
14         // método acessível pois é 'public'
15         System.out.println(carro.toString());
16     }
17 }
```

Modificadores e encapsulamento

Modificadores

- Aplicável: classe, método e atributo
- Descrição: torna o elemento (classe, método ou atributo) acessível para as classes que pertencem ao mesmo pacote
- Obs.: para utilizar este modificador não se deve escrever nenhuma palavra chave

Modificadores e encapsulamento

Modificadores

```
1  package aula05.acesso.garagem;
2
3  class Carro {
4      String marca;
5      String modelo;
6      float velocidade;
7      // construtores
8      Carro() {}
9      Carro(String marca, String modelo) {
10         this.marca      = marca;
11         this.modelo     = modelo;
12         this.velocidade = 0.0f;
13     }
14
15     public String toString() {
16         return "Carro [marca=" + marca + ", modelo=" + modelo
               + ", velocidade=" + velocidade + "]";
17     }
18
19 }
```

Modificadores e encapsulamento

Modificadores

```
1  // local da classe, diferente da classe Carro
2  package aula05.acesso.oficina;
3
4  // não é visivel, pois o modificador é package
5  import aula05.acesso.mPackage.garagem.Carro;
6
7  public class Mecanico {
8      //  não é visivel, pois classe é 'package'
9      Carro carro;
10
11     public void exemplo() {
12         // construtor não é acessível pois é 'package'
13         carro = new Carro("fiat", "mille");
14         // atributo não é acessível pois é 'package'
15         System.out.println(carro.modelo);
16         // método acessível pois é 'public', mas não existe
               objeto criado!
17         System.out.println(carro.toString());
18     }
19 }
```

Modificadores e encapsulamento

Modificadores

```java
// local da classe, igual ao da classe Carro
package aula05.acesso.garagem;

public class Dono {

    Carro carro; // é visivel, pois classe é 'package'

    public void exemplo() {
        // construtor é acessível pois é 'package'
        carro = new Carro("fiat", "mille");
        // atributo é acessível pois é 'package'
        System.out.println(carro.modelo);
        // método acessível pois é 'public'
        System.out.println(carro.toString());
    }

}
```

Modificadores e encapsulamento
Modificadores

Modificador: protected

- Aplicável: método e atributo
- Descrição: torna o elemento (método ou atributo) acessível para as subclasses ou classes que pertencem ao mesmo pacote

Modificadores e encapsulamento

Modificadores

```
1  package aula05.acesso.garagem;
2
3  public class Carro { // protected não é aplicavel em classe
4      protected String marca;
5      protected String modelo;
6      protected float velocidade;
7      // construtores
8      protected Carro() {}
9      protected Carro(String marca, String modelo) {
10         this.marca      = marca;
11         this.modelo     = modelo;
12         this.velocidade = 0.0f;
13     }
14
15     public String toString() {
16         return "Carro [marca=" + marca + ", modelo=" + modelo
               + ", velocidade="+ velocidade + "]";
17     }
18
19 }
```

Modificadores e encapsulamento

Modificadores

```java
// local da classe, diferente da classe Carro
package aula05.acesso.oficina;

// visivel, pois o modificador é public
import aula05.acesso.mProtected.garagem.Carro;

public class Mecanico {
    //  visivel, pois classe é 'public'
    Carro carro;

    public void exemplo() {
        // construtor não é acessível pois é 'protected'
        carro = new Carro("fiat", "mille");
        // atributo não é acessível pois é 'protected'
        System.out.println(carro.modelo);
        // método acessível pois é 'public', mas não existe
            objeto criado!
        System.out.println(carro.toString());
    }
}
```

Modificadores e encapsulamento

Modificadores

```java
// local da classe, igual ao da classe Carro
package aula05.acesso.garagem;

public class Dono {

    // é visivel, pois classe é 'protected'
    Carro carro;

    public void exemplo() {
        // construtor é acessível pois é 'protected'
        carro = new Carro("fiat", "mille");
        // atributo é acessível pois é 'protected'
        System.out.println(carro.modelo);
        // método acessível pois é 'public'
        System.out.println(carro.toString());
    }

}
```

Modificadores e encapsulamento
Modificadores

Modificador: protected

- Como ainda não foi trabalhado o conceito de herança, os exemplos apresentados não diferem do modificador 'package'
- Protected permite que as subclasses, mesmo localizadas em outro pacote, utilizem o elemento (método ou atributo)
- Exemplos serão apresentados em aulas posteriores

Modificadores e encapsulamento

Modificadores

Modificador: private

- Aplicável: método e atributo
- Descrição: torna o elemento (método ou atributo) acessível apenas na própria classe

Modificadores e encapsulamento

Modificadores

```
1  package aula05.acesso.garagem;
2
3  public class Carro { // private não é aplicavel para classe
4      private String marca;
5      private String modelo;
6      private float velocidade;
7      // construtores
8      private Carro() {}
9      private Carro(String marca, String modelo) {
10         this.marca      = marca;
11         this.modelo     = modelo;
12         this.velocidade = 0.0f;
13     }
14
15     public String toString() {
16         return "Carro [marca=" + marca + ", modelo=" + modelo
               + ", velocidade="+ velocidade + "]";
17     }
18
19 }
```

Modificadores e encapsulamento

Modificadores

```java
// local da classe, diferente da classe Carro
package aula05.acesso.oficina;

// visivel, pois o modificador é public
import aula05.acesso.garagem.Carro;

public class Mecanico {
    //  visivel, pois classe é 'public'
    Carro carro;

    public void exemplo() {
        // construtor não é acessível pois é 'private'
        carro = new Carro("fiat", "mille");
        // atributo não é acessível pois é 'private'
        System.out.println(carro.modelo);
        // método acessível pois é 'public', mas não existe
            objeto criado!
        System.out.println(carro.toString());
    }
}
```

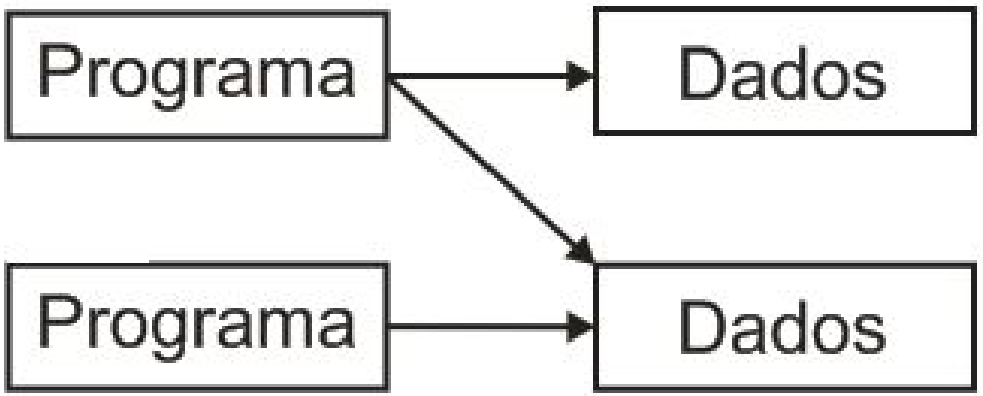
Modificadores e encapsulamento

Modificadores

```
1  // local da classe, igual ao da classe Carro
2  package aula05.acesso.garagem;
3
4  public class Dono {
5
6      // é visivel, pois classe é 'public'
7      Carro carro;
8
9      public void exemplo() {
10         // construtor é acessível pois é 'private'
11         carro = new Carro("fiat", "mille");
12         // atributo é acessível pois é 'private'
13         System.out.println(carro.modelo);
14         // método acessível pois é 'public', mas não existe
               objeto!
15         System.out.println(carro.toString());
16     }
17
18 }
```

Modificadores e encapsulamento
Encapsulamento

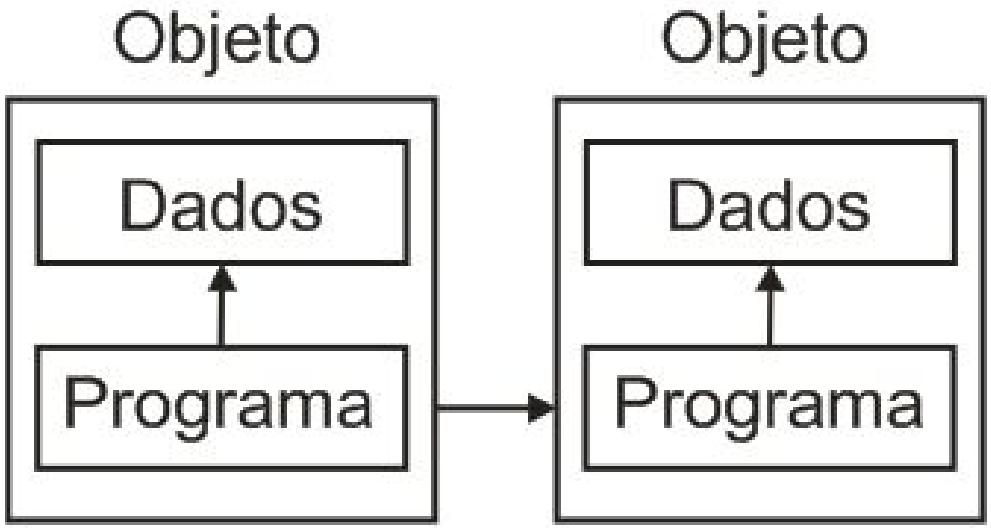
- O conceito de encapsulamento está ligado aos conceitos de *information hiding* (ocultação da informação) e *implementation hiding* (ocultação da implementação).
- Sem encapsulamento qualquer "programa" pode acessar um conjunto de dados.



Modificadores e encapsulamento

Encapsulamento

- O acesso aos dados de uma entidade deve ser restrito
- O desenvolvedor tem controle sobre quais programas podem alterar quais informações.



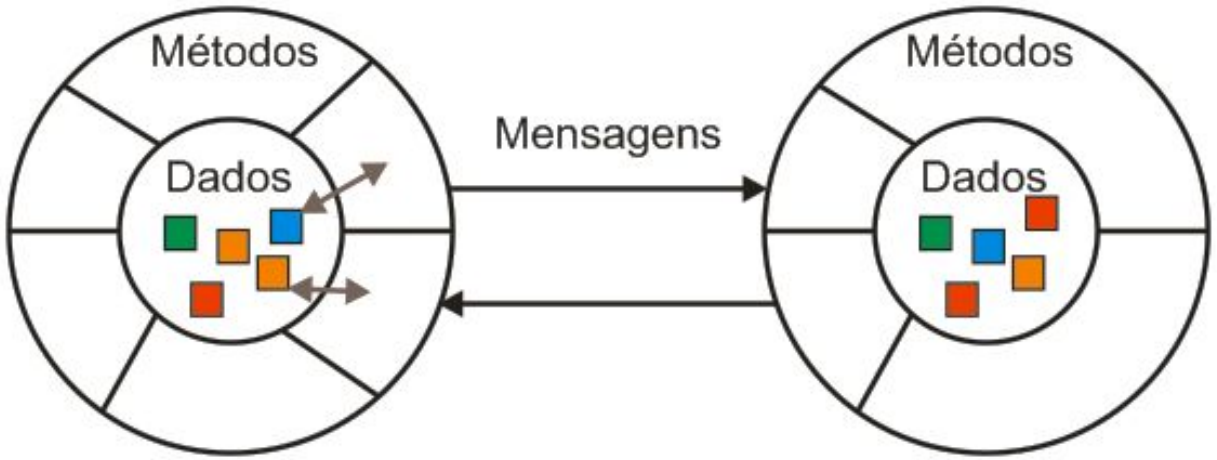
Modificadores e encapsulamento
Encapsulamento

- Para respeitar encapsulamento deve-se definir a interface de uma classe
- Essa interface contém o que é observável/utilizável por outras classes
- Tudo que não faz parte da interface da classe é ocultado

Modificadores e encapsulamento

Encapsulamento

- O acesso aos dados também deve ser restrito:
  - atributos privados;
  - métodos públicos para mutação dos valores dos atributos.



Modificadores e encapsulamento

Encapsulamento

- Por convenção os métodos públicos para mutação dos atributos possuem os nomes:
  - get - para recuperar valores
  - set - para alterar valores
- Cada atributo pode conter um atributo um getAtributo e um setAtributo

```
1  private String nome; // atributo
2
3  // get 'nome do atributo', retorno = tipo do atributo
4  public String getNome() {
5      return nome;  // retorna o valor do atributo
6  }
7
8  // set 'nome do atributo', parametro = tipo do atributo
9  public void setNome(String novoNome) {
10     this.nome = novoNome; // this é uma referencia para o
           objeto
11 }
```

Modificadores e encapsulamento

Encapsulamento

```
1  // identacao para possibilitar apresentacao no slide
2  public class Carro {
3      private String marca;
4      private String modelo;
5      private float velocidade;
6      public Carro() {}
7      public String getMarca() { return marca; }
8      public void setMarca(String marca) { this.marca = marca; }
9      public String getModelo() { return modelo; }
10     public void setModelo(String modelo) { this.modelo =
           modelo; }
11     public float getVelocidade() { return velocidade; }
12     public void setVelocidade(float velocidade) {
           this.velocidade = velocidade; }
13     public String toString() {
14         return "Carro [marca=" + marca + ", modelo=" + modelo
               + ", velocidade="+ velocidade + "]";
15     }
16 }
```

Modificadores e encapsulamento
Encapsulamento

- Os métodos get e set são adotados por convenção, outros nomes podem ser utilizados
- Em uma classe que represente um carro, por exemplo, a velocidade pode ser alterada pelas operações de 'acelerar' e 'frear'

Modificadores e encapsulamento

Encapsulamento

```
1     //... trecho omitido
2
3     private float velocidade;
4
5     public float getVelocidade() {
6         return velocidade;
7     }
8
9     public void acelerar() {
10        this.velocidade += 5;
11    }
12
13    public void frear() {
14        this.velocidade -= 5;
15    }
16
17    //... trecho omitido
```

Modificadores e encapsulamento

Encapsulamento

- Um software para um semáforo que respeite encapsulamento, deve:
  - Restringir o acesso a mudança de seu estado (Verde, Amarelo, Vermelho)
  - Fornecer uma interface que possibilite apenas que as mudanças válidas sejam realizadas

Modificadores e encapsulamento
Modificadores

Modificador: abstract

- Aplicável: Classes e métodos
- Descrição:
  - classes - não permite instanciação e pode conter métodos abstratos (*)
  - métodos - não podem ser implementados (*)

* compreensão depende de conteúdo posterior

Modificadores e encapsulamento

Modificadores

Modificador: final

- Aplicável: Classes, métodos e atributos
- Descrição:
  - classes - não permite que uma classe possua subclasses (*)
  - métodos - não podem ser redefinidos nas subclasses (*)
  - atributos - torna o atributo/variável constante, ou seja não poderá ser alterado

* compreensão depende de conteúdo posterior

Modificadores e encapsulamento

Modificadores

Modificador: final

```
1 public class ModificadorFinal {
2     public static void main(String[] args){
3         final int i = 0;
4         System.out.println(i);
5         // i = 1; // Não pode ser modificada
6         // System.out.println(i);
7     }
8 }
```

Modificadores e encapsulamento

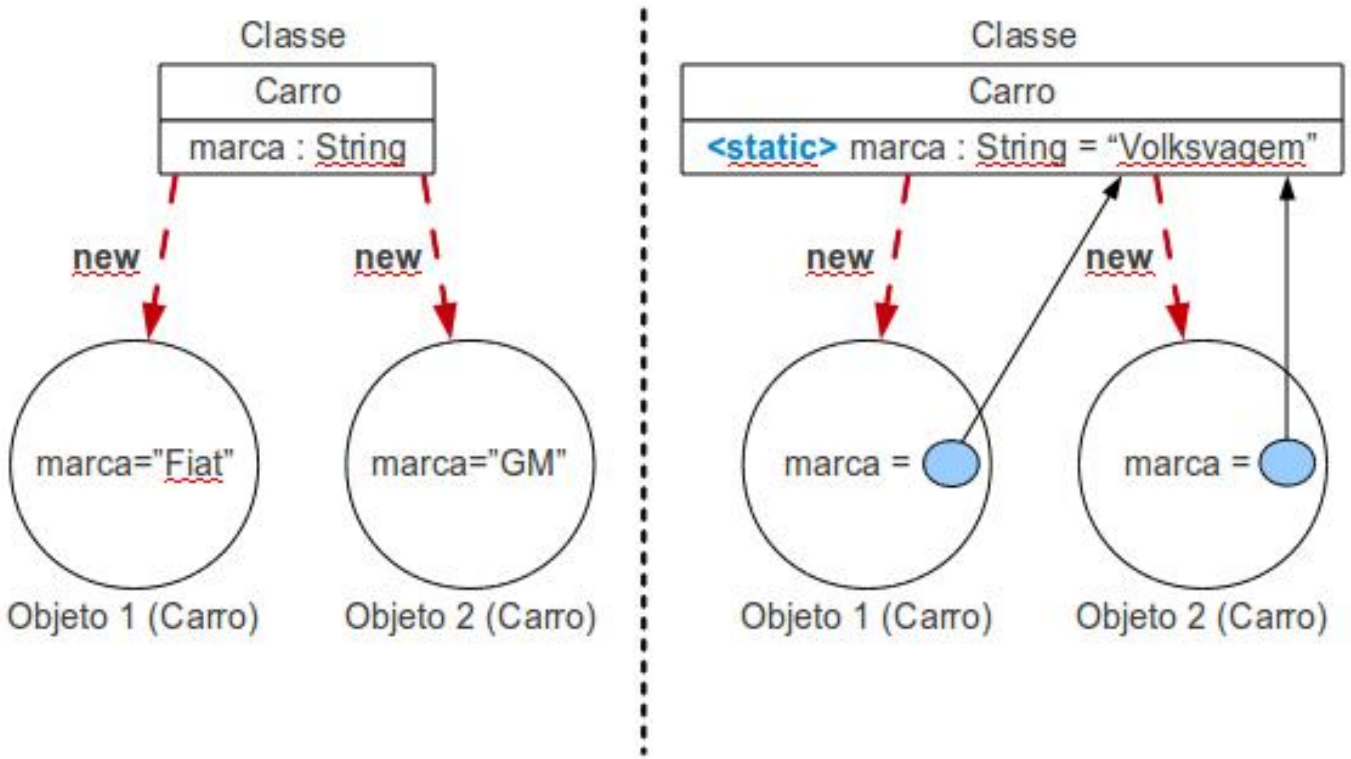
Modificadores

Modificador: static

- Aplicável: Atributos e métodos
- Descrição: torna o elemento (método ou atributo) acessível através da classe, não necessitando de uma instância
- Obs.: Todas as instâncias terão acesso ao mesmo elemento (método ou atributo) da classe.

Modificadores e encapsulamento

Modificadores



Modificadores e encapsulamento

Modificadores

Sem modificador *static*

```
1  public class Carro {
2      String marca; // variável de instância
3  }
```

```
1  public class ModificadorStatic {
2      public static void main(String[] args) {
3          Carro c = new Carro();
4          c.marca = "Fiat";
5          Carro d = new Carro();
6          d.marca = "Chevrolet";
7          System.out.println(c.marca); // Fiat
8          System.out.println(d.marca); // Chevrolet
9      }
10 }
```

Obs.: "marca" pertence a instância, cada objeto guarda um valor

Modificadores e encapsulamento

Modificadores

Modificador: static

```
1 public class Carro {
2     static String marca; // variável de classe
3 }
```

```
1 public class ModificadorStatic {
2     public static void main(String[] args) {
3         Carro.marca = "Volksvagem";
4         Carro c = new Carro();
5         Carro d = new Carro();
6         System.out.println(c.marca); // Volksvagem
7         System.out.println(d.marca); // Volksvagem
8     }
9 }
```

Obs.: "marca"pertence a classe, todos os objetos compartilham o valor

Modificadores e encapsulamento

Modificadores

Modificador: static

```
1  public class ModificadorStatic {
2      public static void main(String[] args) {
3          Carro.marca = "Volksvagem";
4          Carro c = new Carro();
5          Carro d = new Carro();
6          System.out.println(c.marca); // Volksvagem
7          System.out.println(d.marca); // Volksvagem
8          d.marca = "Fiat";
9          System.out.println(c.marca); // Fiat
10         System.out.println(d.marca); // Fiat
11     }
12 }
```

Obs.: alteração pode ser efetuada a partir dos objetos

Modificadores e encapsulamento
Modificadores

- Outros modificadores existentes em Java: synchronized, transient, strictfp, volative e native
- São utilizados com menos frequência que os anteriores

Modificadores e encapsulamento

Modificadores

Modificador: synchronized

- restringe o acesso de threads ao método (apenas uma por vez)

Modificador: transient

- variáveis não serão considerdas na serialização de um objeto

Modificador: strictfp

- membros e variáveis em classes devem seguir o padrão IEEE754 de ponto flutuante

Modificador: volative

- força a sincronização das variáveis de instância (atributos) das cópias das threads com a cópia principal

Modificador: native

- indica o método foi escrito em uma linguagem nativa (ex.: C)

Modificadores e encapsulamento

Exercícios

Exercício

1. Crie uma classe Semaforo obedecendo encapsulamento e as seguintes restrições
   - o estado do semaforo deve ser representado apenas por cores válidas (vermelho, verde ou amarelo)
   - apenas transições válidas devem ocorrer (verde $\rightarrow$ amarelo, amarelo $\rightarrow$ vermelho ou vermelho $\rightarrow$ verde)

Modificadores e encapsulamento

Exercícios

Exercício

1. Crie uma classe Conta obedecendo encapsulamento e as seguintes restrições
   - toda conta possui um número de identificação e um saldo
   - é possível realizar as operações de depósito, saque, e observar o valor do saldo
   - para operações de depósito, não deve ser possível depositar valores negativos
   - para operações de saque, não deve ser possível sacar valores negativos
   - para operações de saque que resultem em um saldo negativo, deve ser cobrada uma taxa de R$ 5,00
   - é possível observar o valor total de taxas que está associado a conta

Modificadores e encapsulamento
Resumo

Resumo

- Modificadores: alteram o significado de determinados elementos
- São os principais modificadores em Java: private, protected, package, public, static, final e abstract
- Encapsulamento: definição de uma interface de classe com acesso controlado aos dados
- Para implementar encapsulamento é necessário restringir o acesso aos atributos, permitindo apenas o acesso controlado a estes
- Métodos GET e SET (mutattors e acessors) são adotados como convenção

Leituras recomendadas

D. J. Barnes e M. Kolling

Programação Orientada a Objetos com Java

Capítulo 2: Entendendo as definições de classe

Seção 3.2 Abstração e modularização

Seção 3.9 Objetos que criam objetos

Seção 5.11: Público versus privado

Seção 5.13: Variáveis de classe e constantes

Rafael Santos

Introdução à programação orientada a objetos usando Java, 2003.

Capítulo 5: Campos e métodos estáticos

Capítulo 10: Pacotes de classes em Java

Leituras recomendadas

FURGERI, S.
Java 6 - Ensino Didático
Capítulo 7: Definições sobre a orientação a objetos

Caelum
Java e Orientação a Objetos, 2011.
Capítulo 6: Modificadores de acesso e atributos de classe

Modificadores e encapsulamento

Perguntas?

Alexandre de Andrade Barbosa
alexandre.barbosa@arapiraca.ufal.br